NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

AN INTEGER PROGRAMMING APPROACH
TO LONG RANGE SHIPBUILDING SCHEDULING

by

Joseph A. Faircloth

September, 1989

Thesis Advisor: Richard E. Rosenthal

Unclassified

Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification: Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report: Approved for public release; distribution is unlimited.
2b Declassification/Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization: Naval Postgraduate School; 6b Office Symbol (*If Applicable*): 55	7a Name of Monitoring Organization: Naval Postgraduate School
6c Address (*city, state, and ZIP code*): Monterey, CA 93943-5000	7b Address (*city, state, and ZIP code*): Monterey, CA 93943-5000
8a Name of Funding/Sponsoring Organization; 8b Office Symbol (*If Applicable*)	9 Procurement Instrument Identification Number
8c Address (*city, state, and ZIP code*)	10 Source of Funding Numbers: Program Element Number; Project No; Task No; Work Unit Accession No

11 Title (*Include Security Classification*)
AN INTEGER PROGRAMMING APPROACH TO LONG RANGE SHIPBUILDING SCHEDULING

12 Personal Author(s) Faircloth, Joseph Anthony

13a Type of Report	13b Time Covered	14 Date of Report (*year, month, day*)	15 Page Count
Master's Thesis	From To	1989, September	46

16 Supplementary Notation: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms (*continue on reverse if necessary and identify by block number*)
Field	Group	Subgroup	Optimization: Integer Programming; Military Applications: Shipbuilding.

19 Abstract (*continue on reverse if necessary and identify by block number*)

This thesis presents an integer programming model to help the Navy develop long-range shipbuilding plans. The model is of a general nature, but is proposed specifically as a decision aid for the developers of the Navy's Extended Planning Annex (EPA). The EPA sets forth planned ship purchases five to 20 years in the future. It is currently produced with a mainly manual process that takes weeks at a time, hence it is extremely difficult for the EPA planners to respond quickly to changes in the given data and assumptions. The optimization model suggests delivery dates for new ships, based on given budgets and requirements, and accounts for such complexities as the extra costs of building a leadship or of resuming construction after a production break. The model has been formulated with the General Algebraic Modeling System (GAMS) and effectively solved with two commercial optimization packages. It performs fast enough to allow the planner to make several 'what if' runs in the course of developing the EPA.

20 Distribution/Availability of Abstract	21 Abstract Security Classification
[X] unclassified/unlimited [ ] same as report [ ] DTIC users	Unclassified

22a Name of Responsible Individual	22b Telephone (*Include Area code*)	22c Office Symbol
Richard E. Rosenthal	(408) 646-2795	55Rl

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted

All other editions are obsolete

security classification of this page

Unclassified

Approved for public release; distribution is unlimited.

An Integer Programming Approach
To Long Range Shipbuilding Scheduling

by

Joseph Anthony Faircloth
Lieutenant, United States Navy
B.S., Freed Hardeman College, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1989

Author: Joseph A. Faircloth
Joseph Anthony Faircloth

Approved By: Richard E. Rosenthal
Richard E. Rosenthal, Thesis Advisor

Siriphong Lawphongpanich
Siriphong Lawphongpanich, Second Reader

Peter Purdue
Peter Purdue, Chairman, Department of Operations Research

ABSTRACT

This thesis presents an integer programming model to help the Navy develop long-range shipbuilding plans. The model is of a general nature, but is proposed specifically as a decision aid for the developers of the Navy's Extended Planning Annex (EPA). The EPA sets forth planned ship purchases five to 20 years in the future. It is currently produced with a mainly manual process that takes weeks at a time, hence it is extremely difficult for the EPA planners to respond quickly to changes in the given data and assumptions. The optimization model suggests delivery dates for new ships, based on given budgets and requirements, and accounts for such complexities as the extra costs of building a leadship or of resuming construction after a production break. The model has been formulated with the General Algebraic Modeling System (GAMS) and effectively solved with two commercial optimization packages. It performs fast enough to allow the planner to make several "what if" runs in the course of developing the EPA.

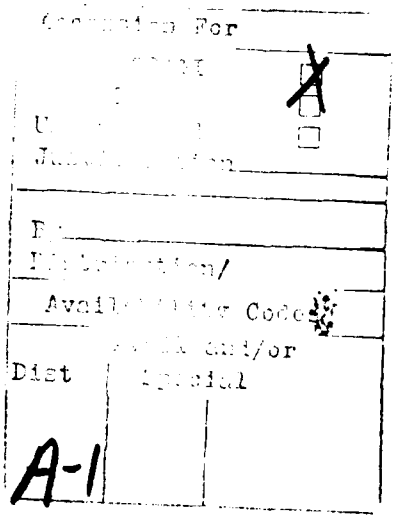

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Sincere thanks go to Professor Terry Harrison of Pennsylvania State University and to Dr. Anthony Brooke of the GAMS Development Corporation, who both made several runs of my model using software (MPSX) unavailable at the Naval Postgraduate School.

Special thanks go to Professor Richard E. Rosenthal of the Naval Postgraduate School for supplying the initial model which was modified and enhanced for this research. This gave me the benefit of not having to "start from scratch" in developing the model.

Thanks also to LCDR Mary Jo Zurey of the Chief of Naval Operations OP-815 (Program Appropriations Branch) staff for contacting Professor Rosenthal requesting assistance with this problem. This provided me with a thesis topic I thoroughly enjoyed.

I. BACKGROUND

This research is an attempt to assist the Navy in planning ship purchases five to 20 years in the future. An optimization model is developed which will produce an initial ship purchase plan. This proposed plan can then be used as a starting point for developing the actual plan set forth in the Extended Planning Annex.

A. EXTENDED PLANNING ANNEX (EPA)

The Extended Planning Annex (EPA) is an annex to the Five Year Defense Program which shows the planned status of the Navy and Marine Corps assets in the years following those covered by the Five Year Defense Program. It is a "best guess" of what the Navy will consist of in those years far in the future. It is not an approved expenditures list, but a long range plan which can be updated or changed as the budget, the current status of the Navy, or its requirements change. The basic concept behind the EPA is an affordability analysis of future programs.

B. HOW THE EPA IS USED

Many decision makers in the Department of the Navy use the EPA in preparing their own long range plans. The EPA shows the projected status of the Navy's ships, aircraft, and personnel based on the Navy's analysts' expectation of what will be affordable. Many decisions concerning today's assets invariably depend on what kinds of assets are expected to be available in the future. For example, if a new carrier is planned in the EPA to come on line in seven years, the personnel who will be rotating at that time will have to decide whether they want to serve on that carrier, and the detailers will be penciling in people to be the high ranking officers. If the new carrier comes on line as scheduled in the EPA the escort ships must be ready to

proceed with the new carrier when it goes out to sea. Also, an older carrier may be sent to the yards for an overhaul based on when the new carrier comes on the line to take its place. This is of course only one example, many others could be cited.

C. HOW THE EPA IS CURRENTLY DEVELOPED

The current procedure for developing the EPA involves a member of the Chief of Naval Operation's OP-81 staff (currently LCDR M. J. Zurey) spending a great amount of time trying to develop a ship purchase schedule which will be affordable and maintain desired fleet characteristics. Some of the scheduling is relatively easy since the decision is mandated by other documents (e.g., the number of ballistic missile submarines is included in treaties). Most decisions have more latitude and are made through a manual trial and error process. Many considerations must be taken into account, and examples of these include:

1. When are ships going to be retiring and thus require replacement?
2. When are ships going to be built to replace retiring ships?
3. How much money in each year is left over after the required ship purchases?
4. Is it better to build a new ship or extend the life of the ships coming up on retirement through the Ship Life Extension Program?
5. Is there enough money to build several ships all of which need to be delivered in the same year? If not, when should each be built?
6. Should the production line for one type of ship be stopped in order to release funds to build another type ship?
7. Will there be enough personnel to man the ships which are being built?
8. Will the budget for ship construction increase or decrease, and by how much?
9. If a ship retires without a replacement, will the Navy be able to meet the national force requirements with the remaining ships?

This is not an exhaustive list of questions and important factors. Fiscal considerations are complex. Ships are not paid for when they are delivered, but require installments years ahead of delivery, and lead ships do not cost the same as a typical ship off the production line. Also, one must remember that allocations of millions or billions of dollars are being planned for use five to twenty years in the future. The need to satisfy many conditions and to keep track of expenditures for several years suggests the use of a computer rather than a manual procedure. The need to make good decisions about how to best allocate money suggests the use of an optimization or simulation model. Optimization or simulation techniques can then be used to assist the decision maker in choosing the best values for the decision variables. Reference 1 describes simulation models as "strategy evaluation models" and optimization models as "strategy generation models." Thus, in order to use a simulation model, every possible combination of purchase plans (or at least every sensible plan) must be generated ahead of time. Each plan is evaluated by the model and the best plan selected. This is simply not feasible with the number of possible alternatives available. This is the reason for choosing an optimization model.

D. THE NEED FOR A COMPUTER MODEL

The current long range plan is developed manually by an analyst at OP-81. One plan generally requires weeks to prepare. If there is a change either in the budget or the threat scenario, the plan must be redeveloped from the beginning. In order to reduce the time required to develop long range plans LCDR Zurey requested that the Naval Postgraduate School's Operations Research Department conduct research on how to computerize the process. As part of the computerization, a model must be developed. The main purpose of this study is the development of an optimization model to aid analysts in preparing long range plans.

II. OPTIMIZATION MODEL

The model is presented in two versions: conceptual and implementation. The conceptual version of the model is designed for ease of presentation, to provide a rough idea of the model's capability before presenting the more complicated version that was actually implemented. Both models use "elastic" [Ref. 2] or "soft" [Ref. 3: pp. 36-37] constraints and a penalty function associated with the elastic variables as a means for finding a highly desirable, if not classically "optimal," long range plan. Elastic variables allow constraints to be violated at a cost. They allow the model to reflect the common managerial practice of violating some constraints in order to satisfy others or to improve the objective function. To accomplish this, two elastic variables are added to each equality constraint, and one elastic variable to each inequality constraint. Elastic variables have penalties as their coefficients in the objective functions. They are restricted to be nonnegative and will take on positive values when the corresponding constraints are violated. One advantage of elastic constraints is the ease of obtaining an initial feasible solution to the model. Following the notation originated in Brown and Graves [Ref. 2], relational operators with a small circle on the top indicate elastic constraints. This notation eliminates the need for explicitly displaying the actual elastic variables in the model, thereby making it easier to read. (The notation also reflects Brown and Gaves' computional practice in their X-System solver [Ref. 4] of treating elastic variables implicitly.)

A. CONSIDERATIONS

To be useful, the model must be realistic and flexible. To be flexible, a model should allow the user to perform "what if" type of analysis (e.g., "What happens to the plan if the projected budget goes down instead of up?", or "What happens to the plan if we need another carrier?"). To be realistic, a model must include the following features:

1. A production line for a ship type may stop and be restarted. This happens when funds must be transferred from one ship type to another. In general, there is also a cost associated with restarting a production line.

2. Allow for bounds on the number of ships being constructed in a given year. Each ship type has an upper bound and a lower bound on the number that can be constructed per year, if any are constructed. There is also an overall bound on the number of ships which may be under construction at any one time.

3. Payments for a ship are required long before the ship is delivered. Some ships require payments in multiple years prior to delivery. Some ships require expenditures after delivery, e.g., for outfitting. Each ship type has its own payment schedule.

4. Leadships require payment structures that are different from the follow-on ships. A leadship costs more and takes longer to build. Additionally, the leadship is delivered in a year by itself, with a year (or perhaps two years) enforced delay before any more ships of that type are delivered.

B. CONCEPTUAL MODEL

1. Index Use

i	Ship type	{ CV, CGN, FFG, ... }
d	Calender year of ship delivery	{ 1990, 1991, 1992, ... }
p	Calender year of payment	{ 1990, 1991, 1992, ... }
k	Ship's production status	{ typical, resumption, leadship }

A "resumption" status indicates that this is the first ship of its type to be constructed following a break in the production line. All the cost of restarting the

production line is added to this ship. The "leadship" status is reserved for the very first ship of a given type. All other ships are given the status of "typical." All inappropriate index combinations are screened out.

2. Given Data

ship quantity data

$need_{id}$ Quantity of ship type i needed in year d.

$have_{id}$ Quantity of ship type i that are in year 0 inventory and will still be in operation in year d.

$getting_{id}$ Quantity of ship type i that are currently under construction and will become available in year d.

The net demand for ship type i up to year d is derived as follows:

$$netdem_{id} = need_{id} - have_{id} - \sum_{d' \le d} getting_{id'}$$

It should be emphasized that the number of ships needed in future years is an input to this model. These needs are determined by other analysts and communicated to the OP-81 officer in charge of scheduling the ship purchases.

fiscal data

c_{ikdp} Cost in year p for each ship of type i and status k delivered in year d.

$budget_p$ Money available to purchase ships in year p.

Fiscal data is generally given in units of millions or hundreds of millions of constant year dollars. Defining the cost parameter with two time subscripts (d and p) allows for greater flexibility in modeling ship costs. For example, if $p < d$ then the cost is due to construction, and if $p > d$ then the cost is due to outfitting. The actual method for inputting the cost is much easier than filling in the four dimensional array c_{ikdp}, as shown in the Appendix.

production limitations data

lag$_i$ Number of years required to construct a ship of type *i*..

upbnd$_{id}$ Upper bound on quantity of ship type *i* that can be delivered in year *d*.

lwbnd$_{id}$ Lower bound on quantity of ship type *i* that can be delivered in year *d* **if any** are delivered.

sybnd$_p$ Upper bound on total number of ships that can be under construction in year *p*.

policy data

Relative penalties for elastic variables associated with elastic constraints. The value assigned to these penalties has a profound influence on the solution. The method used in this research is described in Chapter III.

3. Decision Variables

The decision variables in the conceptual model are:

X_{ikd} Number of ships of type *i* and status *k* to be delivered in year *d*.

Elastic variables associated with elastic constraints also exist, but are not explicitly referred to as decision variables. The value of X_{ikd} for *k* = "resumption" or "leadship" must be binary, while for *k* = "typical" it is a general integer.

4. Formulation

The conceptual formulation minimizes the total penalty cost associated with violation of the elastic constraints to be described below. This type of planning usually involves so many conflicting constraints that it is generally impossible to satisfy all the constraints inelastically.

MIN: $\sum$ PENALTIES

ST: 1) DEMAND$_{id}$: [Ensure that the number of type *i* ships delivered during or prior to year *d* meets the net demand.]

$$\sum_{d' \le d} \sum_{k} X_{ikd'} \doteq \text{netdem}_{id}, \forall\, i,d$$

2) FISCAL$_p$: [Observe budget in year p.]
$$\sum_{ikd} c_{ikdp} X_{ikd} \doteq \text{budget}_p , \forall p$$

3) CONSTRUCT$_p$: [Observe limit on the number of ships under construction in year p.]
$$\sum_{ik} \sum_{0 \le d-p \le lag_i} X_{ikd} \dot{\le} \text{sybnd}_p , \forall p$$

4) Ensure the total number of ship type i delivered in year d is $\in$ { 0, lwbnd$_{id}$, ... , upbnd$_{id}$ }.

5) For new ship types, force the leadship to be delivered before any typical ships.

6) Force a resumption unit to be delivered following a production break.

The first three sets of constraints are relatively easy to handle. The logical constraints (4) - (6) require further development. In particular, constraints (5) and (6) involve non-convex costs, and therefore a linear programming optimizer will tend to violate them. Constraint (4) requires using general integer decision variables with a range of values from a disjoint set of integers (e.g., {0, 3, 4, 5 }). Like the non-convex costs, this condition causes computational difficulty in practice.

C. IMPLEMENTATION MODEL

1. Decision Variables

The implemented version of the model uses binary decision variables exclusively. This choice is more amenable than general integer variables to most solvers and it proves advantageous when constructing logical constraints. However, it increases the number of decision variables. The conceptual model uses index k to distinguish ship status, "typical", "resumption", or "leadship". The implemented version instead uses three separate variables for this purpose.

Additionally, a new index j is used to indicate the quantity ordered for typical ships. The new binary variables and their meaning are:

$XT_{idj} = 1 \Leftrightarrow$ j = number of typical ships of type i to be delivered in year d.

$XR_{id} = 1 \Leftrightarrow$ A resumption ship of type i is to be delivered in year d.

$XL_{id} = 1 \Leftrightarrow$ A leadship of type i is to be delivered in year d.

Since the index k has been dropped, the cost coefficients are similarly redefined:

ct_{idp} Cost in year p for each typical ship of type i delivered in year d.

cr_{idp} Cost in year p for each resumption ship of type i delivered in year d.

cl_{idp} Cost in year p for each leadship of type i delivered in year d.

All inappropriate index combinations are screened out prior to sending the model to the solver. The binary variables with index j are related to the original variables as follows:

$$X_{id'typical'} = \sum_j j \, XT_{idj}$$

A binary variable must be defined for each possible positive value of $X_{id'typical'}$. To allow at most one of the new binary variables to take the value of 1 for each pair of i and d, the following generalized upper bound (GUB) is added:

$$\sum_j XT_{idj} \leq 1, \quad \forall\, i\, d$$

Using this construction of binary variables rather than the usual "powers-of-two" factorization [Ref. 5:p. 190] allows discontinuity to be more easily handled. For example, if $X_{id'typical'} \in \{0, 3, 4, 5\}$ then XT_{idj} will be defined only for j = 3, 4, and 5. The other values will be screened out. It easy to see that j could be defined for any discrete set, even when several discontinuities exist. Handling this with the usual "powers-of-two" factorization would be difficult. The major

drawback to this method is that it is only practical when the number of possible j values is small, otherwise too many binary variables will be introduced.

In the implementation model, we define XT_{idj} for $j = lwbnd_{id}-1, \ldots, upbnd_{id}$. This allows a resumption ship after a production break to be delivered in the same year as $lwbnd_{id}-1$ typical ships.

2. Formulation

The three constraints that were expressed mathematically in the conceptual model translate directly to the implementation model as follows:

$$\text{DEMAND}_{id}: \quad \sum_{d' \le d} \left[\sum_{j} j\, XT_{id'} + XR_{id'} + XL_{id'} \right] \doteq netdem_{id}, \forall\, i\, d$$

$$\text{FISCAL}_{p}: \quad \sum_{i} \sum_{d} \left[\sum_{j} j\, XT_{idj} ct_{idp} + XR_{id} cr_{idp} + XL_{id} cl_{idp} \right] \doteq budget_{p}, \forall\, p$$

$$\text{CONSTRUCT}_{p}: \quad \sum_{i} \sum_{0 \le d-p \le lag_i} \left[\sum_{j} j\, XT_{idj} + XR_{id} + XL_{id} \right] \dot{\le} sybnd_{p}, \forall\, p,$$

provided that the following constraint is also included,

$$\text{GUB}_{id}: \quad \sum_{j} XT_{idj} \dot{\le} 1, \forall\, i\, d.$$

We now describe the implementation of the logical constraints mathematically. The first logical constraint is to ensure the total number of ship type i delivered in year d is $\in \{ 0, lwbnd_{id}, \ldots, upbnd_{id} \}$. The upper bound and lower bound are treated separately.

a. Upper Bound Constraint

Using the binary variables, the upper bound constraint is written simply as:

$$\text{UPPER}_{id}: \quad \sum_{j} j\, XT_{idj} + XR_{id} + XL_{id} \dot{\le} upbnd_{id}, \forall\, i\, d$$

b. Lower Bound Constraint

Ensuring that the number of type i ships delivered in year d is either zero or greater than or equal to the lower bound lwbnd_{id} is the most involved constraint in the model. This constraint needs to be stated explicitly only when $\text{lwbnd}_{id} \geq 2$. (It holds automatically when $\text{lwbnd}_{id} = 1$, because the variables are integer.)

The lower bound constraints involve only typical and resumption ships, because lead ships are always built in isolation. That is, if a leadship is built, the lower and upper bound on total ships of its type is one.

The inequalities designed for handling the lower bound constraint are dependent on knowledge of the constraint's limited scope and on simultaneous enforcement of other constraints. Assume for a specific i,d:

1) $\text{lwbnd}_{id} \geq 2$

2) $\sum_j \text{XT}_{idj} \leq 1$

3) $\text{XL}_{id} = 0 \quad \text{or} \quad \sum_j \text{XT}_{idj} + \text{XR}_{id} = 0$

4) $\text{XT}_{idj} = 0 \text{ for } j \notin \{\text{lwbnd}_{id}\text{-}1, \ldots, \text{upbnd}_{id}\}$

The first assumption is made because otherwise the lower bound constraint under discussion is moot. The second assumption is justified by appealing to the GUB constraint. The third assumption appeals to the "leadship-first" constraint, and the fact that the optimizer will not choose to build a leadship twice, since they cost more. The fourth constraint is justified simply by model design: we define XT_{idj} to exist only for $j \in \{\text{lwbnd}_{id}\text{-}1, \ldots, \text{upbnd}_{id}\}$.

Under these assumptions, there are eight mutually exclusive and collectively exhaustive cases, which can be represented in a four-by-two table. The rows of Table 1 correspond to possible values for the number of typical ships delivered. The columns correspond to possible values for the resumption ships. The entries of the table indicate feasibility with respect to the lower bound constraint.

TABLE 1. FEASIBILITY WITH RESPECT TO THE LOWER BOUND CONSTRAINT.

	Number of Resumption Ships	
Number of Typical Ships	$XR_{id} = 0$	$XR_{id} = 1$
$\sum_j XT_{idj} = 0$	Feasible	Infeasible
$\sum_j XT_{idj} = lwbnd_{id}-1$	Infeasible	Feasible
$\sum_j XT_{idj} = lwbnd_{id}$	Feasible	Feasible
$\sum_j XT_{idj} > lwbnd_{id}$	Feasible	Feasible

Six of the eight cases are feasible and two are infeasible. A set of linear inequalities that correctly discriminates between the feasible and infeasible cases is:

$$XT_{idj} \leq XR_{id} \quad \text{for } j = lwbnd_{id} - 1$$

$$XR_{id} \leq \sum_j XT_{idj}$$

The first of these inequalities rules out the infeasible case represented in the second row, first column of Table 1. The second inequality rules out the other

infeasible case. The feasible cases do not violate either condition. Based on this analysis, we include the following constraints in the implementation model:

$$\text{LOWER1}_{id}: \quad \text{XT}_{idj} - \text{XR}_{id} \stackrel{\cdot}{\leq} 0, \quad j = \text{lwbnd}_{id} - 1, \forall\, i\, d \text{ with } \text{lwbnd}_{id} \geq 2$$

$$\text{LOWER2}_{id}: \quad \text{XR}_{id} - \sum_{j} \text{XT}_{idj} \stackrel{\cdot}{\leq} 0, \quad \forall\, i\, d \text{ with } \text{lwbnd}_{id} \geq 2$$

c. "Leadship-First" Constraints

The next logical constraint to be developed is one that forces a leadship to be purchased before any typical (or resumption) ships of its type are bought. This constraint applies only to new ship types. A zero/one parameter gotsome_i (derived from have_{id} and getting_{id}) indicates whether any ships of type i are in existence or under construction. If $\text{gotsome}_i = 0$, then a set of "leadship-first" constraints for ship type i must be added to the model. Without these constraints, the optimizer would tend to purchase typical ships without ever buying the more expensive leadships.

Assuming binary values for the decision variables and satisfaction of the GUB constraints, the sum

$$\sum_{j} \text{XT}_{idj} + \text{XR}_{id}$$

will have value one or two if any non-leadships of type i are delivered in year d. Otherwise, this sum will be zero.

For any given i,d, with $\text{gotsome}_i = 0$, the sum

$$\sum_{d' < d} \text{XL}_{id'}$$

will be one if a leadship of type i is delivered prior to year d. If no leadship exists in year d, this sum will be zero.

The required constraint is to prevent a non-leadship from being delivered prior to a leadship delivery. This can be expressed as

$$\sum_j XT_{idj} + XR_{id} \leq 2 \sum_{d'<d} XL_{id'}$$

for all $i\ d$ with $\text{gotsome}_i = 0$. In the elastic formulation, the constraint is:

$$\text{LEADFRST}_{id}: \sum_j XT_{idj} + XR_{id} - 2 \sum_{d'<d} XL_{id'} \dot{\leq} 0, \forall\ i\ d \text{ with } \text{gotsome}_i = 0$$

In some Navy planning, delivery of typical ships is not scheduled until two years after the leadship. This affords more time for test and evaluation of the leadship before opening a production line. The model can easily be modified to enforce this scheduling constraint by redefining the leadship-first constraint as

$$\sum_j XT_{idj} + XR_{id} - 2 \sum_{d'<d-1} XL_{id'} \dot{\leq} 0$$

d. Production Resumption Constraints

The next logical constraint from the conceptual model to be formulated mathematically is the requirement to purchase a resumption ship after a production break. Typical ship delivery of type i is allowed in year d only if type i is delivered the previous year or a resumption ship is delivered the same year. That is

$$\sum_j XT_{idj} \geq 1$$

is feasible only if

$$XR_{id} = 1 \text{ or } \sum_j XT_{i\ d-1\ j} + XR_{i\ d-1} + XL_{i\ d-1} \geq 1$$

A constraint which enforces this relationship is:

$$\text{PRODBRK}_{id}: \sum_j XT_{idj} - XR_{id} - \sum_j (XT_{i\ d-1\ j}) - XR_{i\ d-1} - XL_{i\ d-1} \dot{\leq} 0, \forall\ i\ d$$

In order to help keep this constraint from forcing the solver to always lead off with a resumption unit, some early values of XT_{id} and XR_{id} are fixed using the value of $getting_{id}$, if $getting_{id} \geq 1$. Since new ships will require lag_i years to construct, the first lag_i years cannot have any new ships delivered that are not already in construction (and thus are included in $getting_{id}$). The actual method of fixing the decision variables is shown in the GAMS input file in the Appendix. Fixing the decision variables in this way also ensures that the calculated expenditures are correct, and allows the user to input the projected budget, not the projected budget less the amount required for ships already in construction. Additionally, if a leadship is being constructed the user will use a parameter $leadship_{id}$, in the same way as $getting_{id}$. This allows fixing the variable XL_{id} in the same manner. In order to improve solution times the planner may fix XL_{id} to one for some particular year d which is thought to be the year that the leadship will come on line. Then, by varying the fixed delivery year, several solutions may be obtained.

e. Objective/Penalty Function

The objective/penalty function is a function of all the elastic variables multiplied by their penalty. The penalty associated with a particular elastic variable should reflect the ship type and the year associated with the variable. If a constraint must be violated, the user would rather violate constraints associated with smaller ships than with very large and expensive ship. In addition, the user would rather violate constraints as far in the future as possible. With this in mind, the penalties are discounted for ship type and year. The method for discounting by ship type uses the concept that the larger ships are also more expensive. The penalty is multiplied by a ratio of the particular ship's cost to the cost of the most expensive ship. For yearly discounting, the user inputs a value for the parameter discnt. The penalty

discount from one year to the next is (1 + discnt). If the user desires the penalties discounted by 5% per year, discnt will require a value of -0.05. If for some reason the user would rather the model violate constraints in the nearer years (if they must be violated) discnt will require a positive value.

In order to allow the total penalty to be calculated, the penalty factors are scaled to ensure the same units are being added. For example, if the budget constraint is violated, the associated elastic variables need units of dollars (in order to add to or subtract from the budget for that year). But, the other elastic variables will be generally in units of ships. To keep the penalty units the same, the penalty factor associated with the elastic variable for budget violation is divided by the cost of the most expensive ship. This will yield a penalty for budget violation with units of penalty per ship. It should be noted that there is a negative penalty (or reward) for spending less than the budget. The reward for spending less than the budget by some amount is of course much smaller than the penalty for overspending the budget by the same amount. By using this reward system, if two sets of decision variables have the same penalty (other than the reward), the one which uses the least money has the lower overall penalty.

A redundant constraint for XL_{id} similar to the GUB constraint for XT_{idj} is helpful in the integer enumeration. The following constraint ensures that the total number of leadships constructed for ship type i must be no more than one.

$$\text{MAXLEAD}_i: \quad \sum_d XL_{id} \leq 1, \forall\, i, \text{ with gotsome}_i = 0$$

The elastic variables are U1, V1, Y2, Z2, E3, ... , E10, corresponding to the constraint number, with the penalties upen, vpen, ypen, zpen,

pen3, ... , pen10. The implemented formulation (with constraint numbers reassigned) is:

MIN: PENALTY

ST:

1) $\sum_{d' \le d} [\sum_{j} (j\, XT_{id'j}) + XR_{id'} + XL_{id'}] \doteq netdem_{id}, \forall\, i\, d$

2) $\sum_{i\ d} [\sum_{j} (j\, XT_{idj}) ct_{idp} + XR_{id} cr_{idp} + XL_{id} cl_{idp}] \doteq budget_{p}, \forall\, p$

3) $\sum_{i} \sum_{0 \le d-p \le lag_i} [\sum_{j} (j\, XT_{idj}) + XR_{id} + XL_{id}] \dot{\le} sybnd_{p}, \forall\, p$

4) $\sum_{j} (j\, XT_{idj}) + XR_{id} + XL_{id} \dot{\le} upbnd_{id}, \forall\, i\, d$

5) $XT_{idj} - XR_{id} \dot{\le} 0, j = lwbnd_{id} - 1, \forall\, i\, d$ with $lwbnd_{id} \ge 2$

6) $XR_{id} - \sum_{j} (XT_{idj}) \dot{\le} 0, \forall\, i\, d$ with $lwbnd_{id} \ge 2$

7) $\sum_{j} (XT_{idj}) + XR_{id} - 2 \sum_{d' \le d} (XL_{id'}) \dot{\le} 0, \forall\, i\, d$ with $gotsome_i = 0$

8) $\sum_{j} (XT_{idj}) - XR_{id} - \sum_{j} (XT_{i\ d-1\ j}) - XR_{i\ d-1} - XL_{i\ d-1} \dot{\le} 0, \forall\, i\, d$

9) $\sum_{j} XT_{idj} \le 1, \forall\, i\, d$

10) $\sum_{d} XL_{id} \dot{\le} 1, \forall\, i.$ with $gotsome_i = 0$

$$\begin{aligned} \textbf{PENALTY} = & \sum_{id} (upen_{id} U1_{id} + vpen_{id} V1_{id} + pen4_{id} E4_{id} + pen5_{id} E5_{id}) \\ & + \sum_{id} (pen6_{id} E6_{id} + pen7_{id} E7_{id} + pen8_{id} E8_{id} + pen9_{id} E9_{id}) \\ & + \sum_{i} (pen10_{i} E10_{i}) + \sum_{p} (ypen_{p} Y2_{p} - zpen_{p} Z2_{p} + pen3_{p} E3_{p}) \end{aligned}$$

D. OUTPUT

The output from the model is displayed in three reports. An example of partial output reports is displayed below in Figure 1.

PARAMETER REPORT1 SHIPS BUILT VS. SHORTAGE OR EXCESS BY YEAR

	YEAR6 BUILT	YEAR6 SHORT	YEAR6 EXCESS	YEAR7 BUILT	YEAR7 SHORT	YEAR7 EXCESS	YEAR8 BUILT	YEAR8 SHORT	YEAR8 EXCESS
TYPE1	1.00								
TYPE2							1.00		
TYPE3				1.00					
TYPE4		3.00				1.00			
TYPE5	2.00	1.00		4.00	2.00		2.00		
TYPE6		1.00			1.00		1.00		
TYPE7		2.00			2.00			3.00	

PARAMETER REPORT2 EXPENDITURES VS. BUDGET BY YEAR

	YEAR3	YEAR4	YEAR5	YEAR6	YEAR7
EXPENDED	4160.00	4243.00	4329.00	3605.00	2278.00
BUDGET	4160.00	4243.00	4327.00	4413.00	4501.00
OVERRUN			2.00		
SAVINGS				808.00	2223.00

PARAMETER REPORT3 CONSTRAINT ELASTICITY VARIABLES

	E3	E5	E6	E7	E8
TYPE1.YEAR9					0.5
TYPE3.YEAR3	1.0			1.0	
TYPE4.YEAR1		1.0			
TYPE5.YEAR5					0.5
TYPE5.YEAR6					0.5
TYPE6.YEAR9	2.0		1.0		

Figure 1. An example of partial output reports from the model.

The first report, REPORT1, in Figure 1 provides the ship purchasing plan produced by the model. The columns named "BUILT" give the number of ships to be purchased for each ship type during a given year. The columns named "SHORT" and "EXCESS" give the number of ships which are short and in excess, respectively, of the required number of ships specified by the input data called $need_{id}$. With this report the user can either implement the ship purchasing plan under the column "BUILT" for each year and ship type. The more logical option is to use that plan as the starting point and consider the other columns (i.e., "SHORT" and "EXCESS") in order to develop a long range plan that takes other factors not modeled into consideration. For example, if a class of ships is built and the last ship in the class is not built due to upper or lower bounds, then the user may want to order that particular ship with the last order for that ship type, or split the last order.

The idea is to use the model's output as a tool for developing a long range plan, not to expect the model to produce the "optimal" plan. The second report shows the use of money and displays the budget, the amount expended, the amount of budget overrun, and the amount expended below the budget, for each year. The third report displays the values given to the constraint elasticity variables not shown in the first two reports for each time period and year. If the value of some constraint elasticity variable is not zero, then the solution displayed in the first two reports is actually not feasible, with respect to the constraint associated with that elasticity variable. Should the third report show anything other than "all zero", the user should consider modifying the input values associated with elastic penalties, or bounds ($upbnd_{id}$, $lwbnd_{id}$, $sybnd_{d}$, and $budget_{p}$) and rerunning the model. The values to be modified will depend on which elastic variable had a value other than zero. Then, the user should report to their superiors that the stated desires are not feasible and provide them with values which are feasible. This information can also be used as a basis for negotiating new resources or mediating between conflicting desires. The later would normally entail performing sensitivity analysis of the penalties.

E. LIMITATIONS

In our implementation of the model, we have chosen to leave out several realistic features in order to make the model readily understandable. Although these features do represent limitations to the model, they can indeed be included at the expense of increased model and computational complexity. Below, we list these limitations.

1) Except for the extra cost of a leadship or resumption ship, the model assumes the cost associated with a purchase is a linear function of the number of ships purchased. This does not allow quantity discounts, learning-curve effects, or

economies of scale. This deficiency can be corrected, but additional user input and model complexity will be required.

2) There is no lower bound on the total number of ships that can be under construction in a given year.

3) There is no cost associated with shutting down a production line. The cost of the variable XR_{id} includes the cost of restarting a production line, but the cost of shutting down is considered negligible. Associated with this limitation is the assumption that the cost of restarting a production line is not a function of the number of years the production line has been secured.

4) The model does not handle Ship Life Extension Program (SLEP) ships. To do so, constraints which force the solver to consider SLEP ships must be added to the model.

5) The model shown in the Appendix has limits on the magnitude of j set at five and the number of years over which payments may be made for any ship type to five years.

III. COMPUTATIONAL EXPERIENCE

The model is written and tested using the General Algebraic Modeling System, GAMS [Ref. 6]. GAMS is essentially a model-solver interface. It allows the model constraints to be written in an index-exploiting, algebraic form similar to the mathematical form used in scientific communication. This makes it easy to translate the model from the mathematical form presented in the previous chapter to the computer input that can be used by the GAMS interface. GAMS then translates the model into a form required by the solver. The available solvers include ZOOM [Ref. 7] and MPSX [Ref. 8]; both of which are used in this research. According to Reference 9 [page 3], GAMS

1. Provides a high-level language for the compact representation of large and complex models
2. Allows changes to be made in model specifications simply and safely
3. Allows unambiguous statements of algebraic relationships
4. Permits model descriptions that are independent of solution algorithms

Several small scale data sets were used to validate the formulation. Initially, the solution was not required to be integer, and the linear programming solver MINOS [Ref. 10] was used. As the model neared completion, a more realistic sized data set and the ZOOM mixed-integer programming solver were introduced to test the model's ability to generate integer solutions. However, ZOOM "is intended for medium-sized problems with no special structure and up to about 200 zero/one variables." [Ref. 9:p. 225] Since the actual data set requires about four times this many zero/one variables, ZOOM quickly became inadequate with the large model. So, for the purpose of comparing different solvers, an intermediate sized data set was used. This data set allows a maximum of five ships of any one type to be

constructed per year (j), considers ten ship types (i), and has a ten year planning horizon (d). The model was sent to Professor Terry Harrison at Pennsylvania State University through BITNET, an electronic mail network worldwide. At Penn State, the MPSX solver was used to solve the same data set. The results are summarized in Figure 2.

Input data set has :	**Typical Model has:**
10 years (d and p)	335 rows of equations
10 ship types (i)	990 columns of variables
5 ships max per year of each type (j)	268 discrete variables
	4891 non-zero elements
Computer Times:	
NPS IBM 3033AP:	IBM 3090-400:
GAMS: 28.630 sec	GAMS: 9.020 sec
Solver (ZOOM):	Solver (MPSX):
1000 iter's 155.150 sec	10000 iter's 67.800 sec
50000 iter's 1300.068 sec	
Solution quality: 18.9% Gap	Solution quality: 14.6% Gap

Figure 2. Computer time comparison for intermediate sized data set.

The ZOOM solver has difficulty handling even this intermediate sized data set, showing no gain in solution quality from 1000 to 50000 iterations. MPSX, on the other hand, yields a better solution quality much faster and with fewer iterations. Several runs were made with the two solvers as the model continued to develop, with the same results. A larger, more realistic, data set was constructed and solved

with ZOOM and MPSX (via Anthony Brooke), the results are summarized in Figure 3. Interestingly, on this larger problem, both solvers found a better quality solution.

Input data set has :	**Typical Model has:**
15 years (d and p)	1158 rows of equations
20 ship types (i)	2685 columns of variables
5 ships max per year of each type (j)	742 discrete variables
	16688 non-zero elements
Computer Times:	
NPS IBM 3033AP:	IBM 3090-400:
GAMS: 73.520 sec	GAMS: 20.140 sec
Solver (ZOOM):	Solver (MPSX):
30000 iter's 1458.087 sec	24773 iter's 644.400 sec
Solution quality: 5.4% Gap	Solution quality: 0.49% Gap

Figure 3. Computer time comparison for realistic sized data set.

In terms of solution quality, MPSX dominates ZOOM. However, ZOOM does give an integer solution, which is a far better starting point for developing the long range shipbuilding schedule than the typical "don't build anything" starting point. Additionally, in ZOOM's favor, a good solution is generally produced early on in the iteration count (e.g., From Figure 3, the same solution is given after 20000 iterations as is given at 30000 iterations). However, it seems unable to search through this large a problem tree and find an improving solution. ZOOM will run on personal computers (taking much longer than the above times), so it can be used

more easily with the model's actual classified data. MPSX, on the other hand, generally provides a better solution in much less computer time, but requires a mainframe computer.

Another solver option, which is currently being developed for use with GAMS, is the X-System [Ref. 4] which has solved similar problems of a much larger scale in relatively small computer times. Reference 11 develops a very similar model for Army helicopters with 4000 constraints, 21000 variables, 300 binary variables, and 100000 non-zero coefficients which is solved in about a minute when using the X-System (on an IBM 3033AP).

The final option is to discard GAMS as a front end and write an independent solver specific to this model. This would indeed reduce the required computer time, but would do so at great development cost and at the expense of the GAMS flexibility and ease of use. Since the model will probably require solving many times with varied input data sets and several "what if" scenarios, and even with additional constraints, the GAMS front end is too valuable to discard. It is possible, of course, to write a front end to the model specific solver, which is as user friendly as GAMS.

Of particular value throughout this research was the GAMS "dollar operator". The dollar operator is "... used for exception handling in equations." [Ref. 9:p. 92] "A dollar operator within an equation is an implied if-else operation ...." [Ref. 9: p. 94] The dollar operator proved essential in keeping the number of variables and constraints down to a manageable figure. For example, consider the large data set whose features are shown in Figure 3. Without the dollar operator to restrict the variables and constraints there are 2151 constraints, 4551 variables, 2100 of them discrete. This is twice as many constraints and twice as many variables, so the

problem size is four times what it is with the dollar operator. Obviously this feature of GAMS is essential to the model.

A key element of the model is the penalty factors. Setting these penalties correctly is very important. In this study, the penalty factors were set iteratively by solving the linear programming relaxation of the model. This allowed the results to be returned quickly, and did not use up precious computer time. To start with, all penalty factors were set to one, and the general penalty factor for violating logical constraints (epen) set to ten. Then the individual factors were raised and the general penalty lowered to achieve the lowest penalty (with one being the lowest factor used) with a solution which did not violate the logical constraints. The ship balance constraint (DEMAND) was allowed to be violated, and the budget allowed to be under spent, but the other constraints were not allowed to be violated. Examining the marginal value of the elastic variables and constraints was also helpful. The adjustment of penalties continued until the linear program produced a near integer solution at which point the model was transferred to an integer program solver. Since the units are not the same for each constraint, the penalties were scaled so that approximately equivalent units are added in the penalty/objective function. This technique proved to be useful in producing integer solutions, and at setting the penalties.

IV. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

During the course of this study, GAMS has proven to be a valuable tool in the model development. GAMS is both user friendly and quite flexible. It is this flexibility feature that allows for "what if" type analysis. Moreover, one can also develop a "front end" program to interact with GAMS and thus further enhance the user friendliness of the overall model.

As for the integer solvers, two solvers: MPSX and ZOOM were available during the study. Based on a realistically sized data set, MPSX clearly dominates ZOOM. However, ZOOM is available on a 386 based personal computer and MPSX is only available on large mainframe computers. Another solver, the X-System, was not interfaced with GAMS; however, based on reports of its performance on a similar problem, it could conceivably outperform both MPSX and ZOOM.

In Chapter III, it is demonstrated that the model developed in this study does produce the desired result, i.e., an initial plan which analyst/planner can analyze and improve upon. It is cautioned that the model should not be treated as a "black-box" because solutions are "optimal" relative to the data provided by users. In planning, these data are generally rough estimates of actual values, hence the plans produced by the model should be treated at best as a guideline for producing a more sensible plan.

B. RECOMMENDATIONS FOR FURTHER IMPROVEMENT

To further enhance the realism and perhaps the usefulness of the model, the following features can be included in the model.

1. Incorporate Average Ship Age

The helicopter model developed in Reference 11 incorporates average ship age. Not only are new helicopters brought on-line, but that model also determines the retiring times of the current stock of helicopters.

2. Include Shutdown Costs

This model, as presented, does not consider the cost of shutting down the production line, in order to keep the level of complexity compatible with available solvers. A new variable representing the cost of shutting down the production line can be introduced.

3. Use Economies of Scale in Production Costs

As pointed out in the model assumptions, economies of scale when purchasing ships is not allowed. Cost in this model is a linear function of the number of ships purchased of a given type. Since economies of scale exist, their introduction into the cost function could enhance the model.

4. Develop a User-Friendly Front End for the X-System

As stated earlier, the X-System is an alternative solver which is not yet available with GAMS. By developing a front end to the X-System, one would be able to evaluate the advantages of the X-System over the other solvers.

5. Provide Graphical Displays of Solutions

Although displaying solutions to the model numerically is adequate for current usage, a graphical display is more preferable to the user.

APPENDIX : GAMS INPUT FILE

```
$OFFUPPER OFFSYMLIST OFFUELLIST OFFUELXREF OFFSYMXREF
* Long-Range Shipbuilding Scheduling Model

* By LT Joe Faircloth, USN, 8 August 89  (userid 1651p)

* Based on model by Prof. Richard E. Rosenthal,  6 Sep 88

OPTION LIMROW = 0, LIMCOL = 0, SOLPRINT = OFF;
OPTION ITERLIM = 30000, WORK= 15000, RESLIM = 5000;
OPTION OPTCR = 0.00, OPTCA =  1.0;

*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
***  The next area contains data which must be filled   ***
***  in by the user, or a call to a data file with it.  ***
***  The following items must be filled included:       ***
***       Sets: I, T                                    ***
***     Tables: HAVE, GETTING, LEADSHIP, NEED,          ***
***             COST1, COST2, COST3, UPBND, LWBND       ***
***    Scalars: INITBUD, GROWTH, UFAC, VFAC, YFAC       ***
***             DISCNT, EPEN, P3, P4, P5, P6,           ***
***             P7, P8, P9, P10                         ***
*** Parameters: UPBOUND                                 ***
*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***

*$INCLUDE GAMS DATASET A
* NOTE this dataset covers only 5 years, 5 ship types, max of 5 ships
* delivered per year.  This dataset is for demonstration purposes only.

SETS
 I  ship types                     / TYPE1, TYPE2, TYPE3, TYPE4, TYPE5 /
 D  delivery years                 / YEAR1 * YEAR5 /
 J  amount of typical units to buy / 1*5 /
 TIME  time relative to year of delivery   used for cost input only
      / YR-MINUS-0, YR-MINUS-1, YR-MINUS-2, YR-MINUS-3, YR-MINUS-4 /;

TABLE HAVE(I,D) number of each type we have from current inventory
             YEAR1       YEAR2       YEAR3       YEAR4       YEAR5
TYPE1          3           3           3           2           1
TYPE2          2           2           2           1           1
TYPE3          4           4           4           4           3
TYPE4          4           4           3           3           3
TYPE5          0           0           0           0           0

TABLE GETTING(I,D)  units coming online that are in construction now
*                    will come online after year0 but before lag(i)+year1
             YEAR1       YEAR2       YEAR3       YEAR4       YEAR5
TYPE1                      1           2
TYPE2                                  3
TYPE3                      1
```

```
TABLE LEADSHIP(I,D) leadships coming on line that have been started
*                   before year1 and will come on line after year0
             YEAR1       YEAR2       YEAR3       YEAR4      YEAR5
TYPE5                      1

TABLE NEED(I,D)     number of each type we want in each year
             YEAR1       YEAR2       YEAR3       YEAR4      YEAR5
TYPE1          6           6           6           6          6
TYPE2          5           5           5           5          5
TYPE3          8           8           8           8          8
TYPE4          9           9           9           9          9
TYPE5          0           0           0           1          1

TABLE COST1(I,TIME)     typical ship cost data in appropriate year
*                       relative to the delivery year
         YR-MINUS-0  YR-MINUS-1  YR-MINUS-2   YR-MINUS-3   YR-MINUS-4
TYPE1       10                                   100
TYPE2       20                                   200
TYPE3       30                      300
TYPE4       40                                   100          400
TYPE5       50                      500

TABLE COST2(I,TIME)     cost data for first unit after production break
*                       including cost of restarting production line and
*                       the cost of the typical unit
         YR-MINUS-0  YR-MINUS-1  YR-MINUS-2   YR-MINUS-3   YR-MINUS-4
TYPE1       10                                   110
TYPE2       20                                   220
TYPE3       30                      330
TYPE4       40                                   150          400
TYPE5       50                      550

TABLE COST3(I,TIME)     leadship cost data
         YR-MINUS-0  YR-MINUS-1  YR-MINUS-2   YR-MINUS-3   YR-MINUS-4
TYPE5       100                                  800

TABLE UPBND(I,D)     upper bound on the number of each type produced
*                    per year if any are to be produced
                YEAR1 * YEAR15
TYPE1                  2
TYPE2                  5
TYPE3                  1
TYPE4                  4
TYPE5                  2

TABLE LWBND(I,D)     lower bound on the number of each type produced
*                    per year IF ANY are to be produced
                YEAR1 * YEAR15
TYPE1                  2
TYPE2                  3
TYPE3                  1
TYPE4                  3
TYPE5                  1
```

```
PARAMETER UPBOUND(D) upper bound on the total ships under construction;
*                    in year d
 UPBOUND(D) = 10;

SCALARS  INITBUD  first year budget                  / 2000/
         GROWTH   budget growth rate as percent      / 0.02/ ;

* Set penalty factors for constraint violations
SCALARS
   DISCNT time weighting factor for penalties                 /-0.05/
   UFAC   ship shortage factor per ship                       /2/
   VFAC   ship excess factor per ship                         /2/
   YFAC   budget overrun factor per cost of largest ship      /10/
   EPEN   general constraint violation penalty                /10/
   P3     penalty for violating overall construction bound    /1/
   P4     penalty for violating upper bound per ship          /1/
   P5     penalty for violating lower bound                   /1/
   P6     penalty for making only one ship if LT lwbnd        /1/
   P7     penalty for not producing leadship first            /1/
   P8     penalty for not making a resumption unit if reqd    /2/
   P9     penalty for violating GUB bound on XT               /1/
   P10    penalty for violating GUB bound on XL               /1/;

*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
***  End of area containing data which must be filled   ***
***  in by the user.  The next area has aborts to       ***
***  ensure the data has been entered correctly.        ***
*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***

ABORT $(SUM((I,D)$(HAVE(I,D) LT 0),1) GT 0)
"***=>Data entry error in table HAVE, entries may not be negative",
 HAVE;

ABORT $(SUM((I,D)$(GETTING(I,D) LT 0),1) GT 0)
"***=>Data entry error in table GETTING, entries may not be negative",
 GETTING;

ABORT$(SUM((I,D)$((LEADSHIP(I,D) LT 0) OR (LEADSHIP(I,D) GT 1)),1) GT 0)
"***=>Data entry error in table LEADSHIP, entries may only be zero or
one",
 LEADSHIP;

ABORT $(SUM((I,D)$(NEED(I,D) LT 0),1) GT 0)
"***=>Data entry error in table NEED, entries may not be negative",
 NEED;

ABORT $(SUM((I,D)$(NEED(I,D) LT (HAVE(I,D)+GETTING(I,D))),1) EQ CARD(I)
 *CARD(D)) "***=>NEED is satisfied by HAVE+GETTING, for every I,D",
 "    There is no need to run the program, it is already optimal",
 HAVE,GETTING,NEED;

ABORT $(SUM((I,TIME)$(COST1(I,TIME) LT 0),1) GT 0)
 "***=>Data entry error in table COST1, all costs must be non-negative",
 COST1;
```

```
ABORT $(SUM((I,TIME)$(COST2(I,TIME) LT 0),1) GT 0)
 "***=>Data entry error in table COST2, all costs must be non-negative",
 COST2;

ABORT $(SUM((I,TIME)$(COST3(I,TIME) LT 0),1) GT 0)
 "***=>Data entry error in table COST3, all costs must be non-negative",
 COST3;

ABORT $(SUM((I,D)$(UPBND(I,D) LT 0),1) GT 0)
"***=>Data entry error in table UPBND, entries may not be negative",
 UPBND;

ABORT $(SUM((I,D)$(LWBND(I,D) LT 0),1) GT 0)
 "***=>Data entry error in table LWBND, entries may not be negative",
 LWBND;

ABORT $(SUM((I,D)$(UPBND(I,D) LT LWBND(I,D)),1) GT 0)
 "***=>Data entry error in table LWBND or UPBND, LWBND must be less",
 "     than or equal to UPBND, for all i,d.",LWBND,UPBND;

ABORT $(INITBUD LE 0)
 "***=>Data entry error for INITBUD, entry must be positive",
 INITBUD;

ABORT $((UFAC LT 0) OR (VFAC LT 0) OR (YFAC LT 0))
"***=>Data entry error in UFAC,VFAC,or YFAC, values may not be
negative",
 UFAC,VFAC,YFAC;

ABORT $((UFAC + VFAC + YFAC) EQ 0)
 "***=>All penalties are zero, program is optimal as is.",
 "     UFAC, VFAC, and YFAC must not all be zero.", UFAC,VFAC,YFAC;

*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
***      End of ABORTS checking user data input.        ***
***      The next areacalculates additional data        ***
***      not required to be input by the user.          ***
*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***

* D is used for year of delivery and P is used for year of payment
ALIAS (D,P);

PARAMETER CT(I,D,P), CR(I,D,P), CL(I,D,P);
CT(I,D,P)$(ORD(D) - ORD(P) EQ 0) = COST1(I,"yr-minus-0");
CT(I,D,P)$(ORD(D) - ORD(P) EQ 1) = COST1(I,"yr-minus-1");
CT(I,D,P)$(ORD(D) - ORD(P) EQ 2) = COST1(I,"yr-minus-2");
CT(I,D,P)$(ORD(D) - ORD(P) EQ 3) = COST1(I,"yr-minus-3");
CT(I,D,P)$(ORD(D) - ORD(P) EQ 4) = COST1(I,"yr-minus-4");

CR(I,D,P)$(ORD(D) - ORD(P) EQ 0) = COST2(I,"yr-minus-0");
CR(I,D,P)$(ORD(D) - ORD(P) EQ 1) = COST2(I,"yr-minus-1");
CR(I,D,P)$(ORD(D) - ORD(P) EQ 2) = COST2(I,"yr-minus-2");
CR(I,D,P)$(ORD(D) - ORD(P) EQ 3) = COST2(I,"yr-minus-3");
CR(I,D,P)$(ORD(D) - ORD(P) EQ 4) = COST2(I,"yr-minus-4");
```

```
CL(I,D,P)$(ORD(D) - ORD(P) EQ 0) = COST3(I,"yr-minus-0");
CL(I,D,P)$(ORD(D) - ORD(P) EQ 1) = COST3(I,"yr-minus-1");
CL(I,D,P)$(ORD(D) - ORD(P) EQ 2) = COST3(I,"yr-minus-2");
CL(I,D,P)$(ORD(D) - ORD(P) EQ 3) = COST3(I,"yr-minus-3");
CL(I,D,P)$(ORD(D) - ORD(P) EQ 4) = COST3(I,"yr-minus-4");

PARAMETER LAG(I);
LAG(I)$(COST1(I,"yr-minus-4") GT 0) = 4;
LAG(I)$(COST1(I,"yr-minus-3") GT 0 AND LAG(I) EQ 0) = 3;
LAG(I)$(COST1(I,"yr-minus-2") GT 0 AND LAG(I) EQ 0) = 2;
LAG(I)$(COST1(I,"yr-minus-1") GT 0 AND LAG(I) EQ 0) = 1;
LAG(I)$(COST1(I,"yr-minus-0") GT 0 AND LAG(I) EQ 0) = 0;

PARAMETER  BUDGET(P)  money available in year P to purchase ships;
           BUDGET(P)$(ORD(P) EQ 1) = INITBUD;
           LOOP (P,BUDGET(P+1) = FLOOR((1+GROWTH)*BUDGET(P)));

PARAMETER  DISCOUNT(D) discount factor for time weighting penalties;
 DISCOUNT(D)  =  POWER(1+DISCNT,ORD(D)-1);

* Calculate the penalty values for each unit of excess, shortage,
* and budget overrun for each type and year. Calculate all elastic
* constraint penalties.  All penalties are discounted by year and
* relative to ship type cost.
* so that the right hand sides of the dual equations will be integers.
PARAMETERS
   BIGC,PRIORITY(I),UPEN(I,D),VPEN(I,D),YPEN(D),ZPEN(D),PEN3(D),
   PEN4(I,D),PEN5(I,D),PEN6(I,D),PEN7(I,D),PEN8(I,D),PEN9(I,D),PEN10(I);
  BIGC        = SMAX(I,SUM(TIME,COST1(I,TIME)));
  PRIORITY(I)= SUM(TIME,COST1(I,TIME)) / BIGC;
  UPEN(I,D)   = UFAC * DISCOUNT(D) * PRIORITY(I);
  VPEN(I,D)   = VFAC * DISCOUNT(D) * PRIORITY(I);
  YPEN(D)     = YFAC * DISCOUNT(D)/BIGC;
  ZPEN(D)     = YPEN(D)/10;
  PEN3(D)     = EPEN*P3*DISCOUNT(D);
  PEN4(I,D)   = EPEN*P4*DISCOUNT(D)*PRIORITY(I);
  PEN5(I,D)   = EPEN*P5*DISCOUNT(D)*PRIORITY(I);
  PEN6(I,D)   = EPEN*P6*DISCOUNT(D)*PRIORITY(I);
  PEN7(I,D)   = EPEN*P7*DISCOUNT(D)*PRIORITY(I);
  PEN8(I,D)   = EPEN*P8*DISCOUNT(D)*PRIORITY(I);
  PEN9(I,D)   = EPEN*P9*DISCOUNT(D)*PRIORITY(I);
  PEN10(I)    = EPEN*P10*PRIORITY(I);

PARAMETER GOTSOME(I) equals 1 if you do not need to make a lead ship;
  GOTSOME(I) = 0 + 1$(SUM(D,HAVE(I,D)+GETTING(I,D)+LEADSHIP(I,D)) GT 0);

PARAMETER NETDEM(I,D) totl number of i ships that must be del'd by yr d;
  NETDEM(I,D) = NEED(I,D) - HAVE(I,D);

*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
***       All data is now calculated. The next area      ***
***       defines the variables and the constraints.     ***
*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
```

```
VARIABLES
    XT(I,D,J) order j of type i to be delivered in year d
    XR(I,D)   first one after a break in production costs more
    XL(I,D)   first one of type i to come on line costs more
    U1(I,D)   shortage of type i in year d
    V1(I,D)   excess of type i in year d
    Y2(P)     amount expenditure is over budget in year p
    Z2(P)     amount expenditure is under budget in year p
    E3(D)     elasticity variable for equation construct
    E4(I,D)   elasticity variable for equation upper
    E5(I,D)   elasticity variable for equation lower1
    E6(I,D)   elasticity variable for equation lower2
    E7(I,D)   elasticity variable for equation leadfrst
    E8(I,D)   elasticity variable for equation prodbrk
    E9(I,D)   elasticity variable for equation gub
    E10(I)    elasticity variable for equation maxlead
    PENTOT    total penalties;

BINARY VARIABLES   XT,XR,XL;

POSITIVE VARIABLES   U1,V1,Y2,Z2,E3,E4,E5,E6,E7,E8,E9,E10;

*set x variables to getting(i,t) for those that occur prior to
*year1+lag(i). This may prevent making a prod break type when not needed
*and will allow the correct budget amount tc be used.
   XT.FX(I,D,J)$(((GETTING(I,D-1) GE 1) OR (LEADSHIP(I,D-1) EQ 1))
        AND (ORD(J) EQ GETTING(I,D))) = 1;
   XR.FX(I,D)$(((GETTING(I,D-1) EQ 0) AND (LEADSHIP(I,D-1) EQ 0))
        AND (GETTING(I,D) GE 1)) = 1;
   XT.FX(I,D,J)$(((GETTING(I,D-1) EQ 0) AND (LEADSHIP(I,D-1) EQ 0))
        AND ((GETTING(I,D) GE 2) AND (ORD(J) EQ GETTING(I,D) - 1))) = 1;
   XL.FX(I,D)$(LEADSHIP(I,D) EQ 1) = 1;

*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
***     Define dynamic sets of variables for use        ***
***     with dollar operators in the constraints        ***
*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***

SETS POS1(I,D), POS2(J,I,D), POS3(I), POS4(I,D), POS5(I,D,J), POS6(D,P),
     POS7(I,D), POS8(I,D,P), POS9(I,D), POS10(I,D), POS11(I,D,P),
     POS12(I,D,J), POS13(I,D), POS14(I,D), POS15(J,I,D);

 POS1(I,D)   =YES$(ORD(D) GT LAG(I));
 POS2(J,I,D) =YES$((ORD(J) GE LWBND(I,D)-1) AND (ORD(J) LE UPBND(I,D)));
 POS3(I)     =YES$(GOTSOME(I) EQ 0);
 POS4(I,D)   =YES$((GOTSOME(I) EQ 0) AND (ORD(D) GT LAG(I)));
 POS5(I,D,J) =YES$(ORD(J) EQ LWBND(I,D)-1);
 POS6(D,P)   =YES$(ORD(P) LE ORD(D));
 POS7(I,D)   =YES$((ORD(D) GT LAG(I)) AND (LWBND(I,D) GE 2));
 POS8(I,D,P) =YES$((ORD(P) LT ORD(D)) AND (ORD(P) GT LAG(I)));
 POS9(I,D)   =YES$((ORD(D) GT LAG(I)) OR (GETTING(I,D) GT 0));
 POS10(I,D)  =YES$(((GOTSOME(I) EQ 0) AND (ORD(D) GT LAG(I)))
                                  OR (LEADSHIP(I,D) EQ 1));
 POS11(I,D,P)=YES$((ORD(D)-ORD(P) GE 0) AND (ORD(D)-ORD(P) LE LAG(I))
                    AND ((GETTING(I,D) GT 0) OR (ORD(D) GT LAG(I))));
```

```
 POS12(I,D,J)=YES$(((ORD(J) GE LWBND(I,D)-1) AND (ORD(J) LE UPBND(I,D)))
                    AND ((ORD(D) GT LAG(I)) OR (GETTING(I,D) GT 0)));
 POS13(I,D)  =YES$((ORD(D)-1 GT LAG(I)) OR (GETTING(I,D-1) GT 0));
 POS14(I,D)  =YES$(((GOTSOME(I) EQ 0) AND (ORD(D)-1 GT LAG(I)))
                     OR (LEADSHIP(I,D-1) EQ 1));
 POS15(J,I,D)=YES$(((ORD(J) GE LWBND(I,D)-1) AND (ORD(J) LE UPBND(I,D)))
                   AND ((ORD(D)-1 GT LAG(I)) OR (GETTING(I,D-1) GT 0)));

*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
***            Begin listing the constraints            ***
*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***

EQUATIONS
  DEMAND(I,D)  maintain inventory balance
  FISCAL(P)    observe budget limits
  CONSTRUCT(D) prevents making more than the ship yard can make
  UPPER(I,D)   total type i made in period t is less than max
  LOWER1(I,D)  with next eqn prevents making less than lwbnd(i t)
  LOWER2(I,D)  with previous eqn prevents making less than lwbnd(i t)
  LEADFRST(I,D) ensures you make a first one before others if needed
  PRODBRK(I,D) ensures next type i after a prod break is prod break type
  GUB(I,D)     ensures only one of the x binary types is made  GUB
  MAXLEAD(I)   gub bound on lead ships
  OBJDEF;

DEMAND(I,D)$POS1(I,D) ..
   SUM(P$POS6(D,P),SUM(J$POS12(I,P,J),
   ORD(J)*XT(I,P,J))
    + XL(I,P)$POS10(I,P)
    + XR(I,P)$POS9(I,P))
   =E= NETDEM(I,D) - U1(I,D) + V1(I,D);

FISCAL(P) ..
  SUM((I,D), SUM(J$POS12(I,D,J),
  XT(I,D,J)*ORD(J))*CT(I,D,P)
 +XL(I,D)$POS10(I,D)*CL(I,D,P)
 +XR(I,D)$POS9(I,D)*CR(I,D,P))
  =E= BUDGET(P) + Y2(P) - Z2(P);

CONSTRUCT(P) ..
  SUM(I, SUM(D$POS11(I,D,P),
  SUM(J$POS2(J,I,D), ORD(J)*XT(I,D,J))
  + XR(I,D) + XL(I,D)$POS10(I,D))) =L= UPBOUND(P) + E3(P);

UPPER(I,D)$POS1(I,D) ..
   SUM(J$POS2(J,I,D), XT(I,D,J)*ORD(J)) + XR(I,D)
   + XL(I,D)$POS10(I,D) =L= UPBND(I,D) + E4(I,D);

LOWER1(I,D)$POS7(I,D) ..
   SUM(J$POS5(I,D,J),XT(I,D,J)) - XR(I,D) =L= 0 + E5(I,D);

LOWER2(I,D)$POS7(I,D) ..
  XR(I,D) - SUM(J$POS2(J,I,D), XT(I,D,J)) =L= 0 + E6(I,D);
```

```
LEADFRST(I,D)$POS4(I,D) ..
   SUM(J$POS2(J,I,D), XT(I,D,J)) + XR(I,D)
   - SUM(P$POS8(I,D,P), XL(I,P)*2)
   =L= 0 + E7(I,D);

PRODBRK(I,D)$POS1(I,D) ..
   SUM(J$POS2(J,I,D),
   XT(I,D,J)) - XR(I,D) - XR(I,D-1)$POS13(I,D)
   - XL(I,D-1)$POS14(I,D)
   - SUM(J$POS15(J,I,D), XT(I,D-1,J)) =L= 0 + E8(I,D);

GUB(I,D)$POS1(I,D) ..
   SUM(J$POS2(J,I,D), XT(I,D,J)) =L= 1 + E9(I,D);

MAXLEAD(I)$POS3(I) ..
   SUM(D$POS1(I,D), XL(I,D)) =L= 1 + E10(I);

OBJDEF ..
   SUM((I,D)$POS1(I,D), UPEN(I,D)*U1(I,D) + VPEN(I,D)*V1(I,D))
 + SUM(P, YPEN(P)*Y2(P) - ZPEN(P)*Z2(P)) + SUM(D, PEN3(D)*E3(D))
 + SUM((I,D)$POS1(I,D), XRPEN*XR(I,D) + E4(I,D)*PEN4(I,D)
 + E5(I,D)*PEN5(I,D) + E6(I,D)*PEN6(I,D) + E9(I,D)*PEN9(I,D)
 + E7(I,D)*PEN7(I,D) + E8(I,D)*PEN8(I,D) )
 + SUM(I$POS3(I),E10(I)*PEN10(I)) =E= PENTOT;

*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
*** All variables and constraints are now defined. Next ***
*** send the model to the solver, then display the      ***
*** solution from the solver.                           ***
*** *** *** *** *** *** *** *** *** *** *** *** *** *** ***

MODEL SHIPS /ALL/;
SOLVE SHIPS USING  MIP MINIMIZING PENTOT;

PARAMETER REPORT1(I,D,*) ships built vs. shortage or excess by year;
  REPORT1(I,D,"BUILT") =SUM(J,XT.L(I,D,J)*ORD(J)) +XR.L(I,D) +XL.L(I,D);
  REPORT1(I,D,"SHORT") = U1.L(I,D);
  REPORT1(I,D,"EXCESS")= V1.L(I,D);
OPTION REPORT1:2:1:2;
DISPLAY REPORT1;



PARAMETER REPORT2(*,P) expenditures vs. budget by year;
  REPORT2("EXPENDED",P) = BUDGET(P) - Z2(P) + Y2(P);
  REPORT2("BUDGET",P)   = BUDGET(P);
  REPORT2("OVERRUN",P)  = Y2.L(P);
  REPORT2("SAVINGS",P)  = Z2.L(P);
OPTION REPORT2:2;
DISPLAY REPORT2;
```

```
PARAMETER REPORT3  constraint elasticity variables;
 REPORT3(I,D,"E3")$(ORD(I) EQ 1) = E3.L(D);
 REPORT3(I,D,"E4")  = E4.L(I,D);
 REPORT3(I,D,"E5")  = E5.L(I,D);
 REPORT3(I,D,"E6")  = E6.L(I,D);
 REPORT3(I,D,"E7")  = E7.L(I,D);
 REPORT3(I,D,"E8")  = E8.L(I,D);
 REPORT3(I,D,"E9")  = E9.L(I,D);
 REPORT3(I,D,"E10")$(ORD(D) EQ 1) = E10.L(I);

OPTION REPORT3:1:2:1;
DISPLAY REPORT3;
```

LIST OF REFERENCES

1. Bradley, Stephen P., Hax, Arnoldo C., and Magnanti, Thomas L., *Applied Mathematical Programming*, pages 6-7, Addison-Wesley Publishing Company, 1977.

2. Brown, G. G. and Graves, G. W., "Elastic Programming: A New Approach to Large-Scale Mixed Integer Optimization," paper presented at the ORSA/TIMS meeting, Las Vegas, Navada, 17 November, 1975.

3. Liebman, J., Lasdon, L., Schrage, L., and Waren A., *Modeling and Optimization with GINO*, The Scientific Press, 1986.

4. Brown, G. G. and Graves, G. W., "Design and Implementation of a Large-Scale (Mixed-Integer) Optimization System," *ORSA/TIMS*, Las Vegas, Navada, November, 1975.

5. Schrage, Linus, *Linear, Integer and Quadratic Programming with LINDO*, The Scientific Press, 1986.

6. Bisschop, J. and Meeraus, A., "On the Development of a General Algebraic Modeling System in a Strategic Planning Environment," *Mathematical Programming Studies*, Vol. 20, 1982.

7. Singal, Jaya., Marsten, Roy E., and Morin, Thomas, "Fixed Order Branch-and-Bound Methods for Mixed-Integer Programming: The ZOOM System," working paper, Management Information Science Department, The University of Arizona, Tucson, Arizona, December, 1987.

8. IBM, International Business Machines Corporation, *Mathematical Programming System Extended (MPSX) and Generalized Upper Bounding (GUB) Program Description*, IBM Manual SH20-0968-1, White Plains, New York, 1972.

9. Brooke, Anthony, Kendrick, David, and Meeraus, Alexander, *GAMS: User's Guide*, The Scientific Press, 1988.

10. Murtagh, Bruce A., and Saunders, Michael A., "MINOS 5.1 User's Guide," Report SOL 83-20R, Stanford University, December 1983, revised January, 1987.

11. Brown, G., Clemence, Robert D., Teufert, William R., and Wood, Kevin R., "An Optimization Model for Army Helicopter Fleet Modernization," Technical Report, Naval Postgraduate School, Monterey, California, 1989.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chief of Naval Operations OP-81, Navy Dept Washington, D.C., 20350-2000	5
4.	Prof. Richard E. Rosenthal, Code 55Rl Operations Research Department Naval Postgraduate School Monterey, California, 93943-5000	5
5.	Prof. Siriphong Lawphongpanich, Code 55Lp Operations Research Department Naval Postgraduate School Monterey, California, 93943-5000	1
6.	Prof. Gerald G. Brown, Code 55Bn Operations Research Department Naval Postgraduate School Monterey, California, 93943-5000	1
7.	Prof. Terry Harrison Management Sciences Department Pennsylvania State University University Park, Pennsylvania, 16803	1
8.	Dr. Anthony Brooke 1925 N. Van Buren Arlington, Virginia, 22205	1
9.	LCDR M. J. Zurey Chief of Naval Operations OP-815A3, Navy Dept Washington, D.C., 20350-2000	1

10. LT Joseph A. Faircloth 5
Department Head Class 111
Surface Warfare Officers School Command
Newport, Rhode Island, 02841-5012